分布式会话跟踪系统架构设计与实践

张志桐@美团基础架构中心 20160625

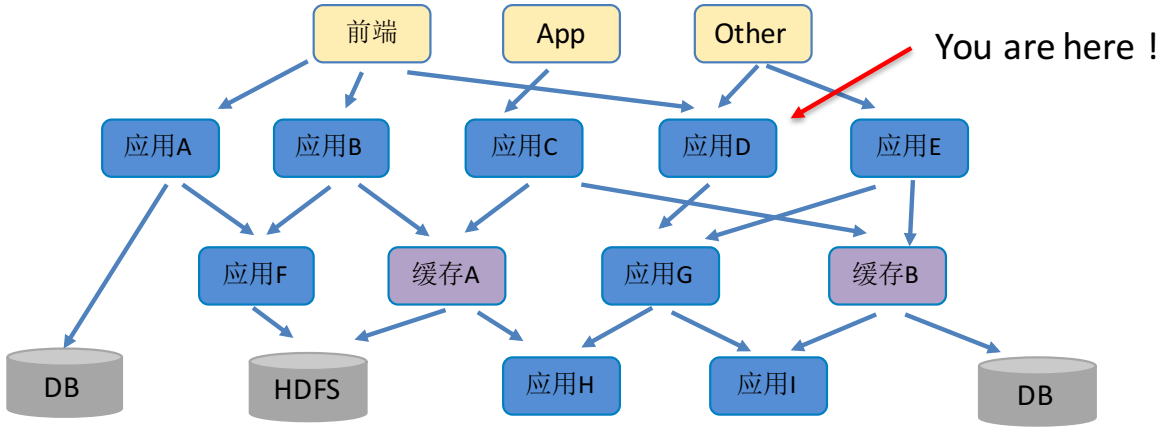
What How Why

- mtrace 项目介绍
- mtrace 使用场景
- mtrace 设计实现

Mtrace

背景

- 日益复杂的分布式系统
- 梳理服务之间的调用关系
- 不同中间件之间相互调用
- 业务感知不到上下游关系



Example

- 警察抓小偷
  - 13:45 小偷去便利店买些道具
  - 14:00 小偷入室行窃并逃逸
  - 18:00 小偷买了开往xxx地的车票
  - 20:00 小偷在xxx酒店入住
  - 20:10 小偷给家里打了个电话
  - 23:00 小偷被抓获

破案关键-身份证

- 你买了些什么商品，银行流水
- 你去过哪里，买了到哪的车票
- 你的电话号是多少
- 你什么时间在哪住过酒店

Example

通过**身份证号**将犯罪轨迹串联起来

- 123456 在xx超市买了xxx
- 123456 在xx路上行窃后逃逸
- 123456 买了去xxx地的车票
- 123456 住入了xx酒店

是否也可以通过一个**id**将请求串联起来

- ? 服务A调用了服务B
- ? 服务B调用了服务C, 出现结果异常
- ? 服务C调用了服务D
- ? 服务D调用了服务E

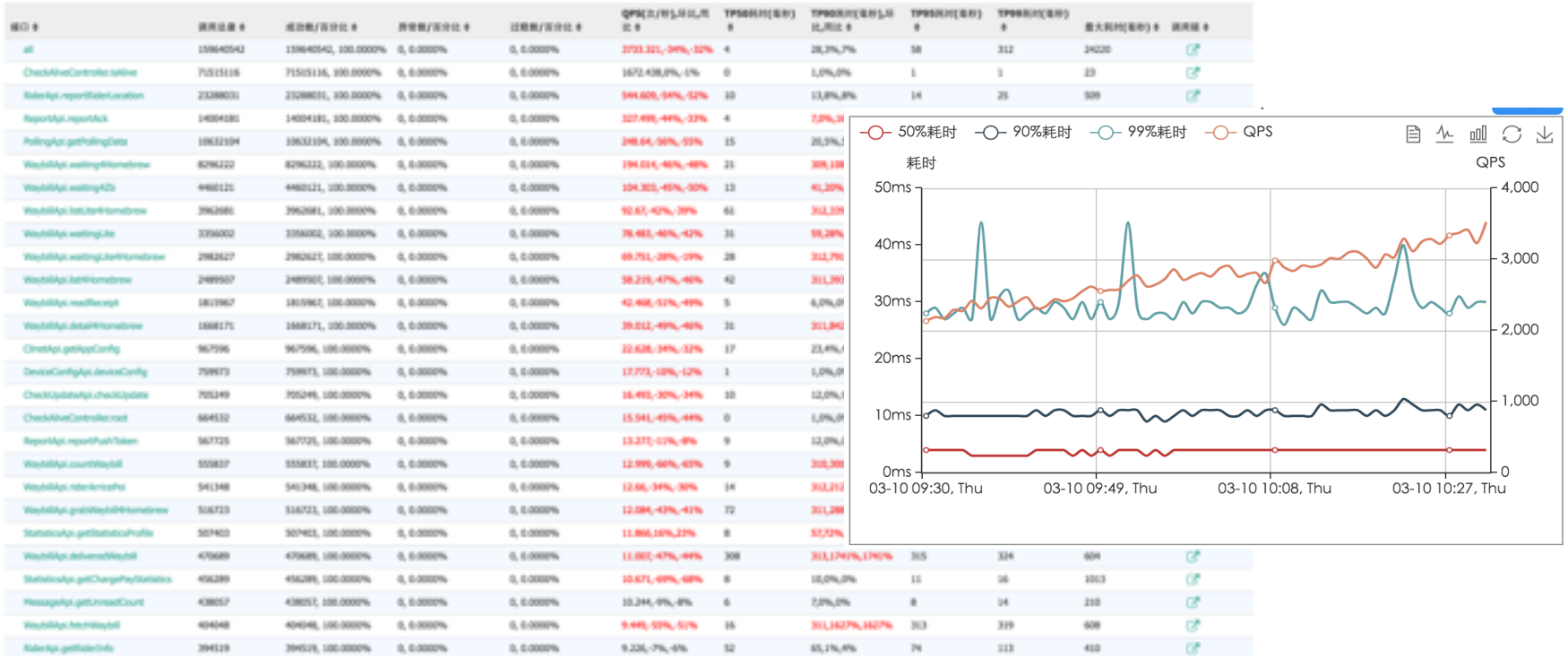
项目介绍

- Mtrace 分布式会话跟踪系统
- 调用链
  - 通过全局id将分布在各个服务上的一次请求串联起来,还原调用关系,追踪问题,分析数据
- 各维度业务指标统计
  - 统计数据, 业务服务QPS, TP数据, 可用性

站在巨人的肩膀上

- Dapper, Google
  - Google 2010 年发布dapper论文
- Zipkin, Twitter
  - Zipkin dapper的开源实现
- EagleEye, Taobao
  - 阿里丰富调用链所做的事情

性能分析-统计



50%耗时
90%耗时
99%耗时
QPS
耗时
QPS
50ms
40ms
30ms
20ms
10ms
0ms
4,000
3,000
2,000
1,000
0
03-10 09:30, Thu
03-10 09:49, Thu
03-10 10:08, Thu
03-10 10:27, Thu

调用链路

调用链面板，展示了该次调用所经过的各个服务的概况信息

服务间层级关系，服务名，具体ip，耗时等信息

调用链路图，各个服务的调用情况

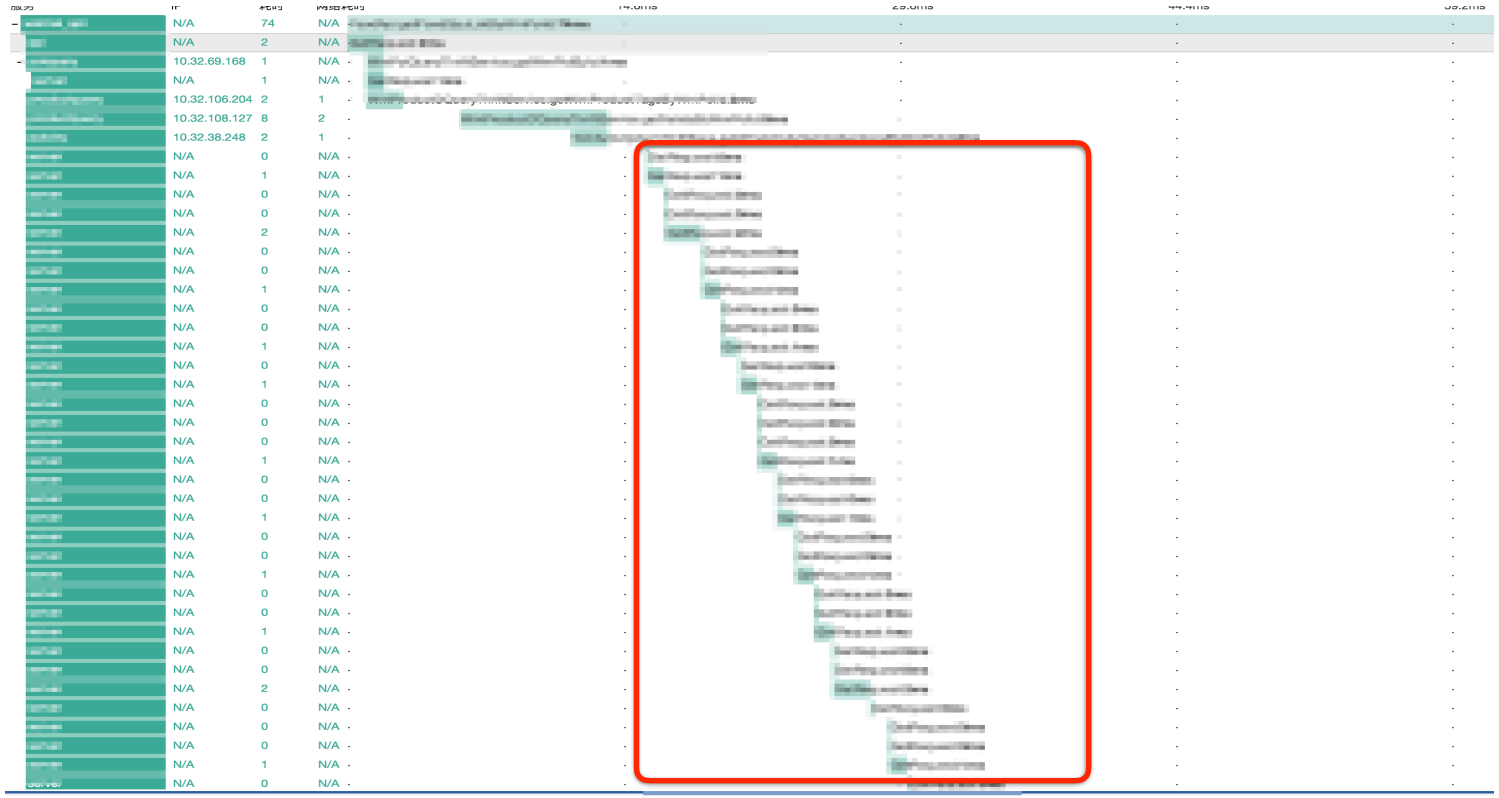
请求路由

通过实际的调用ip查询跨机房调用等情况

链路分析

定位系统瓶颈服务

链路分析

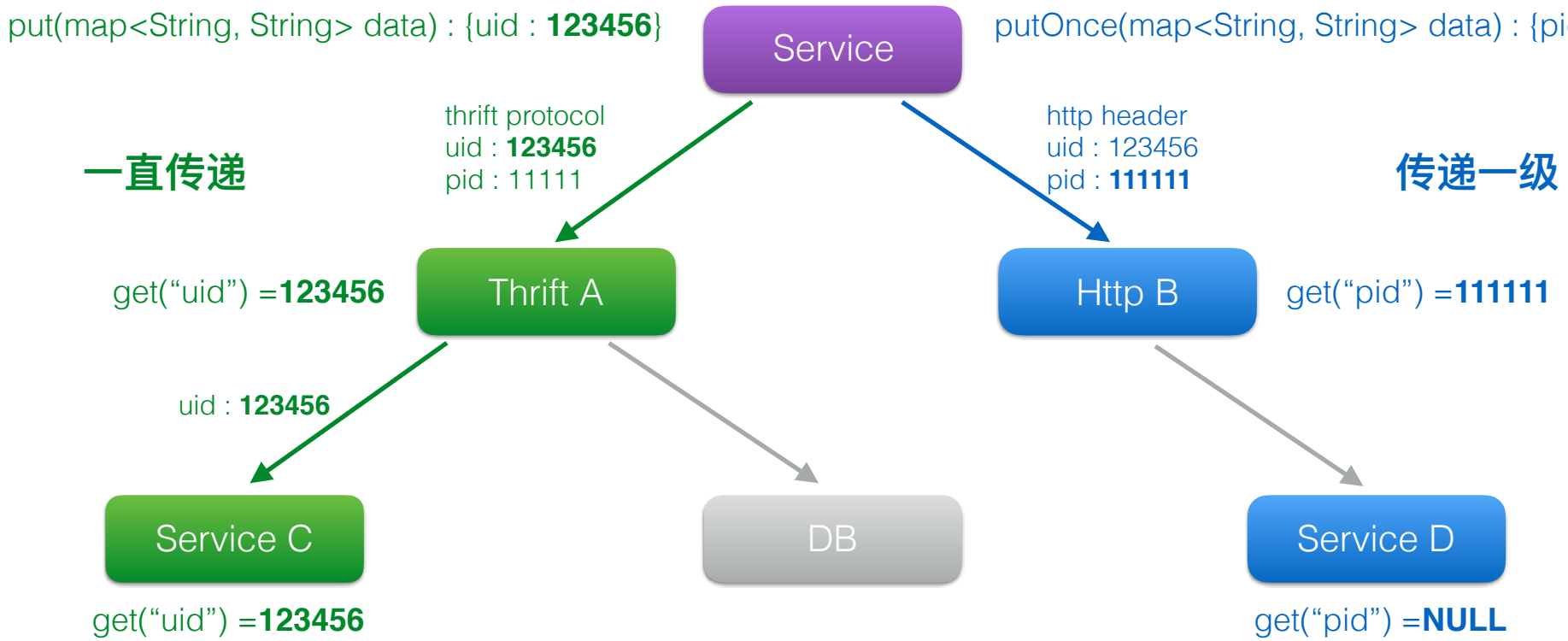


提高系统并行度或优化系统调用

链路分析

- 以前
  - 业务查log，需要登录N台机器mapping请求，定位问题
- 现在
  - 业务log 会打印当前请求的traceId，通过调用链定位超时等异常问题
  - 关联整个链路上不同服务间的异常栈

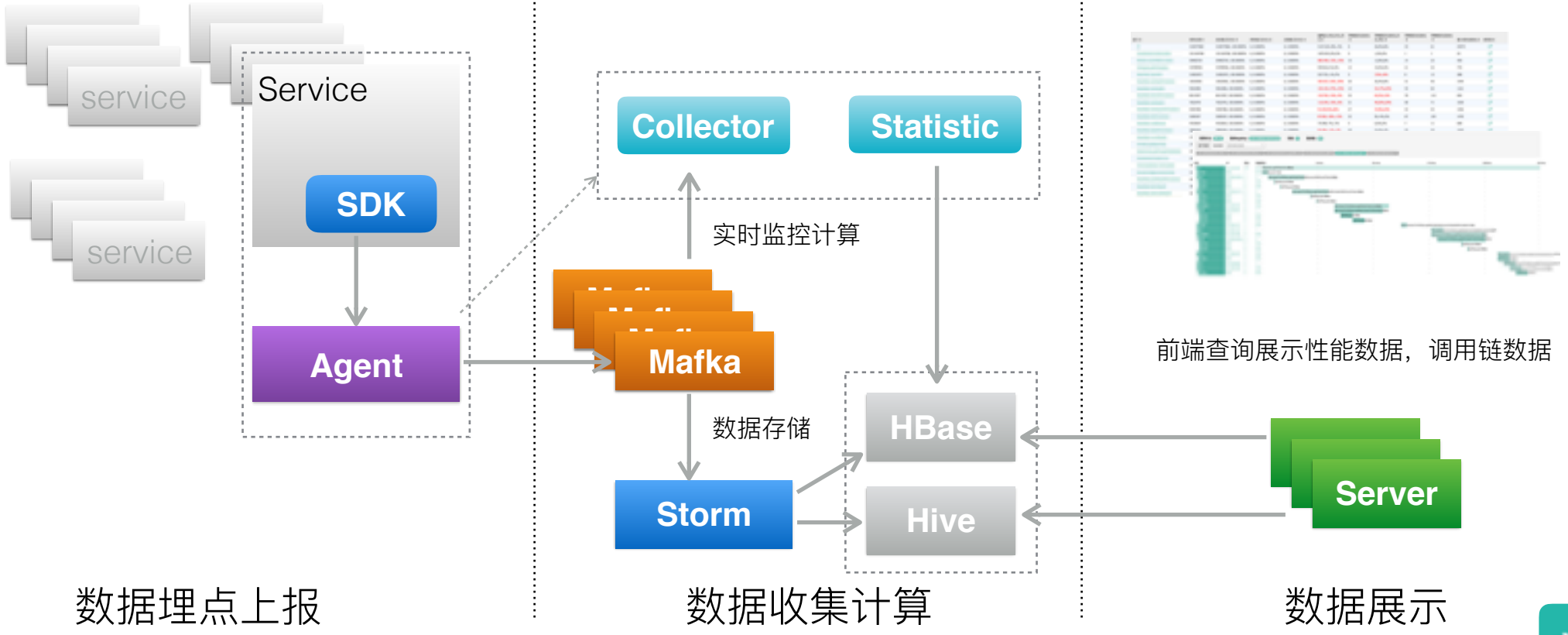
透明传输数据


put(map<String, String> data) : {uid : 123456}
Service
putOnce(map<String, String> data) : {pid : 111111}
thrift protocol
uid : 123456
pid : 11111
http header
uid : 123456
pid : 111111
一直传递
传递一级
get("uid") =123456
Thrift A
Http B
get("pid") =111111
uid : 123456
Service C
DB
Service D
get("uid") =123456
get("pid") =NULL

透明传输数据

- Mtrace 自身数据
  - traceId, spanId, sample. debug
- 业务自定义数据
  - KV 结构
  - 路由策略控制： 通过参数控制后端处理的逻辑选择
  - 调试指令　　： 通过调试指令进行一些业务log的追踪
  - 临时数据　　： 临时传递数据

系统架构


service
service
Service
SDK
Agent
Collector
Statistic
实时监控计算
Mafka
数据存储
Storm
HBase
Hive
Server
前端查询展示性能数据，调用链数据
数据埋点上报
数据收集计算
数据展示

数据埋点

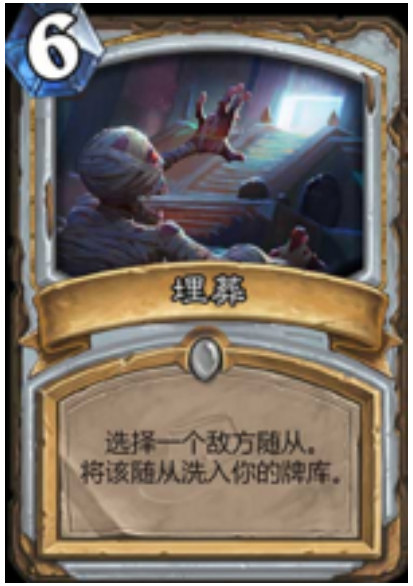
- SDK
  - 埋点，生成调用上下文
  - 同步调用上下文存放在ThreadLocal, 异步调用通过显式调用支持
  - 网络中传输关键埋点数据，用于数据传递，支持thrift, http协议
- Agent
  - 透传数据，用作数据转发
  - 做流量控制

数据埋点

- 埋点中间件
  - RPC 服务： mtthrift
  - Http 服务 : mtrace-http
  - Mysql : mtrace-zebra
  - Cache 缓存 : tair
  - Mq : mafka client

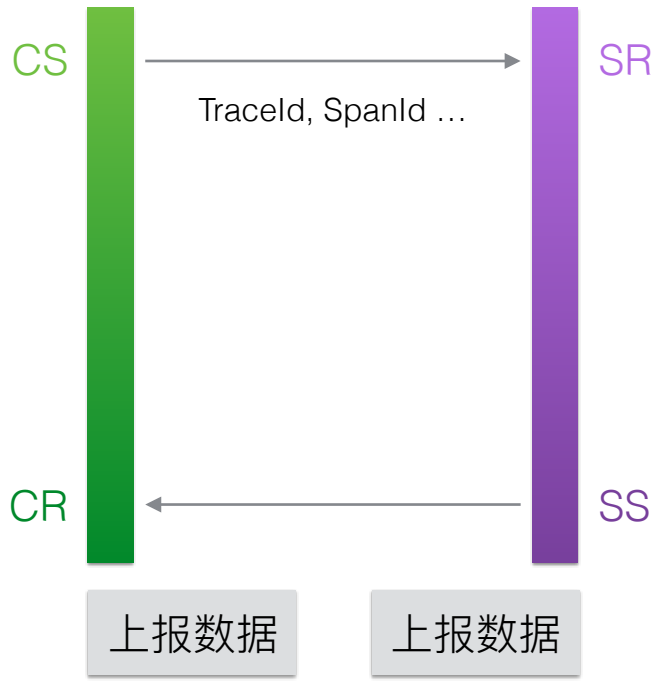
数据埋点

- TraceId
  - 全局唯一标识，64位整数
- SpanId
  - 签名方式生成：0, 0.1, 0.1.2, 0.2 …
- Annotation
  - 业务自定义埋点，uid, 订单id等关键信息
- Other
  - 服务名，方法名，耗时，结果，中间件类型等



数据埋点-四个阶段

1. Client Send
客户端发起请求
生成调用上下文



2. Server Recieve
服务端接收请求并生成上下文

3. 业务端自定义埋点自己感兴趣的数据

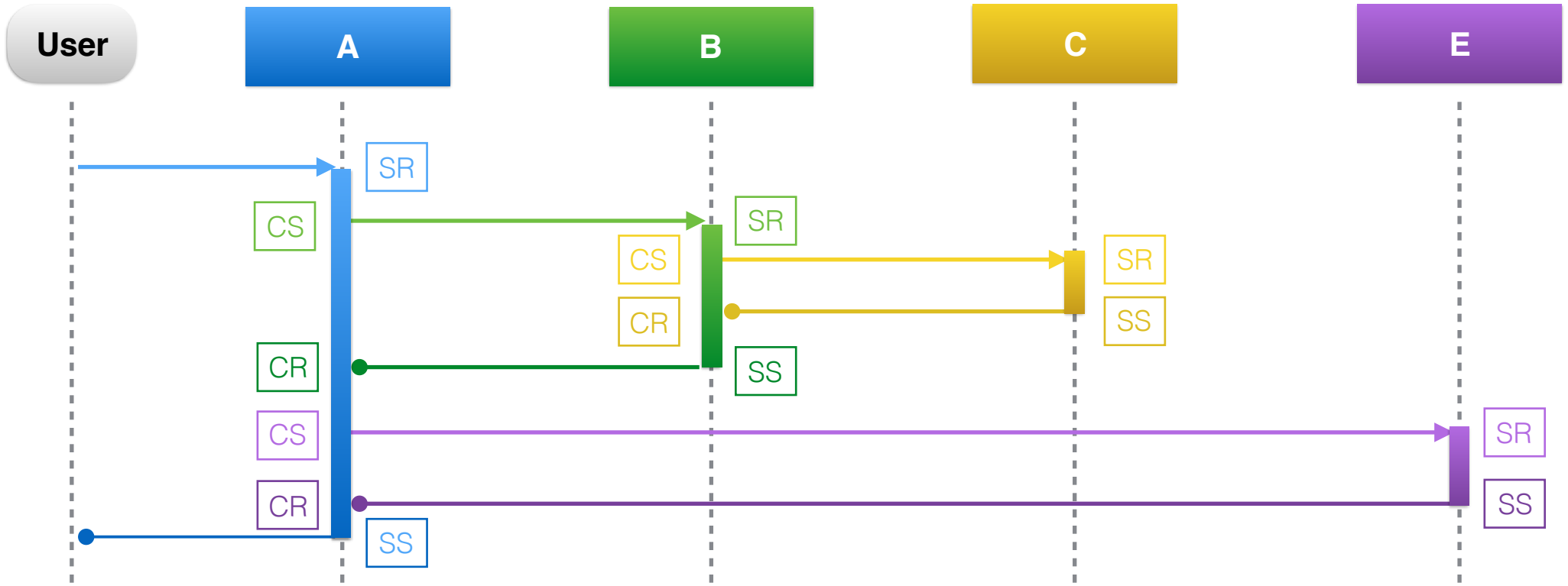
4. Server Send
服务端返回请求结果

5. Client Recieve
客户端接收返回结果

异步队列上报client端上下文数据

异步队列上报server端上下文数据

数据埋点－上下文



CS, SR : 创建上下文 **CR, SS** : 归档上下文

数据埋点－上下文归档

traceId 123456, spanId 0.1.1, appKey C, method C.method, start 106, duration 30, side server

traceId 123456, spanId 0.1.1, appKey B, method C.method, start 105, duration 33, side client

traceId 123456, spanId 0.1 , appKey B, method B.method, start 103, duration 38, side server

traceId 123456, spanId 0.1 , appKey A, method B.method, start 103, duration 38, side client

traceId 123456, spanId 0.2 , appKey E, method E.method, start 148, duration 12, side server

traceId 123456, spanId 0.2 , appKey A, method E.method, start 146, duration 15, side client

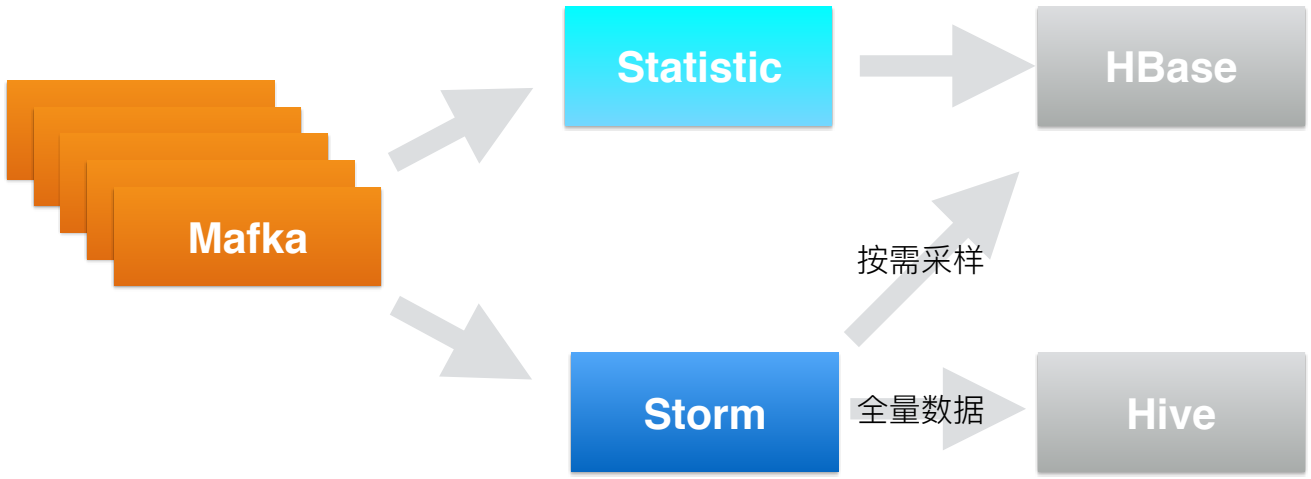
traceId 123456, spanId 0 , appKey U, method A.method, start 100, duration 82, side server

数据埋点－遇到的问题

- 异步调用
  - 异步IO造成的线程切换，不能通过ThreadLocal传递上下文
  - 显式的通过api进行埋点传递，切换前保存，切换后还原
- 数据量大，每天千亿级别的调用
  - 批量上报
  - 数据压缩
  - 极端情况下采样

数据存储

- 实时数据HBase
  - 用于实时调用链路查询
  - 性能统计数据
- 离线数据Hive
  - 离线分析
  - 定制化查询汇总



调用链实时数据存储

- 实时数据Hbase，用于实时调用链路查询
- Rowkey traceId 天然随机
- 列名 SpanId - Type, 埋点上下文

	T									
RowKey	0-C	0.1-S	0.1.1-C	0.1.1-S	0.1.2-C	0.1.2-S	0.2-C	0.2-S	0.2.1-C	0.2.1-S
traceId1	Span	Span	Span	Span						
traceId2	Span	Span					Span	Span	Span	Span
traceId3	Span	Span	Span	Span	Span	Span	Span	Span		

前端展示

谢谢大家